## AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

## Listing of Claims:

1      1.    (Original) A method for executing a commit instruction to
2  facilitate transactional execution on a processor, comprising:
3      encountering the commit instruction during execution of a program,
4  wherein the commit instruction marks the end of a block of instructions to be
5  executed transactionally; and
6      upon encountering the commit instruction, successfully completing
7  transactional execution of the block of instructions preceding the commit
8  instruction;
9      wherein changes made during the transactional execution are not
10  committed to the architectural state of the processor until the transactional
11  execution successfully completes.

1      2.    (Original) The method of claim 1, wherein successfully completing
2  the transactional execution involves:
3      atomically committing changes made during the transactional execution;
4  and
5      resuming normal non-transactional execution.

1      3.    (Original) The method of claim 2, wherein atomically committing
2  changes made during the transactional execution involves:

3    treating store-marked cache lines as locked, thereby causing other

4  processes to wait to access the store-marked cache lines;

5    clearing load marks from cache lines;

6    committing store buffer entries generated during transactional execution to

7  memory, wherein committing each store buffer entry involves unmarking, and

8  thereby unlocking, a corresponding store-marked cache line; and

9    committing register file changes made during transactional execution.


1    4.    (Original) The method of claim 1, wherein if an interfering data

2  access from another process is encountered during the transactional execution and

3  prior to encountering the commit instruction, the method further comprises:

4    discarding changes made during the transactional execution; and

5    attempting to re-execute the block of instructions.


1    5.    (Original) The method of claim 1, wherein for a variation of the

2  commit instruction, successfully completing the transactional execution involves:

3    atomically committing changes made during the transactional execution;

4  and

5    commencing transactional execution of the block of instructions following

6  the commit instruction.


1    6.    (Original) The method of claim 1, wherein potentially interfering

2  data accesses from other processes are allowed to proceed during the transactional

3  execution of the block of instructions.


1    7.    (Original) The method of claim 1, wherein the block of

2  instructions to be executed transactionally comprises a critical section.


4

1       8.     (Original) The method of claim 1, wherein the commit instruction

2  is a native machine code instruction of the processor.

1       9.     (Original) The method of claim 1, wherein the commit instruction

2  is defined in a platform-independent programming language.

1      10.    (Original) A computer system that supports a commit instruction to

2  facilitate transactional execution, wherein the commit instruction marks the end

3  of a block of instructions to be executed transactionally, comprising:

4        a processor; and

5        an execution mechanism within the processor;

6        wherein upon encountering the commit instruction, the execution

7  mechanism is configured to successfully complete transactional execution of the

8  block of instructions preceding the commit instruction;

9        wherein changes made during the transactional execution are not

10  committed to the architectural state of the processor until the transactional

11  execution successfully completes.

1      11.    (Original) The computer system of claim 10, wherein while

2  successfully completing transactional execution, the execution mechanism is

3  configured to:

4        atomically commit changes made during the transactional execution; and

5  to

6        resume normal non-transactional execution.

1      12.    (Original) The computer system of claim 11, wherein while

2  atomically committing changes made during the transactional execution, the

3  execution mechanism is configured to:

4    treat store-marked cache lines as locked, thereby causing other processes

5 to wait to access the store-marked cache lines;

6    clear load marks from cache lines;

7    commit store buffer entries generated during transactional execution to

8 memory, wherein committing each store buffer entry involves unmarking, and

9 thereby unlocking, a corresponding store-marked cache line; and to

10    commit register file changes made during transactional execution.


1    13.  (Original) The computer system of claim 10, wherein if an

2 interfering data access from another process is encountered during the

3 transactional execution and prior to encountering the commit instruction, the

4 execution mechanism is configured to:

5    discard changes made during the transactional execution; and

6    attempt to re-execute the block of instructions.


1    14.  (Original) The computer system of claim 10, wherein if a variation

2 of the commit instruction is encountered, the execution mechanism is configured

3 to:

4    atomically commit changes made during the transactional execution; and

5 to

6    commence transactional execution of the block of instructions following

7 the commit instruction.


1    15.  (Original) The computer system of claim 10, wherein the computer

2 system is configured to allow potentially interfering data accesses from other

3 processes to proceed during the transactional execution of the block of

4 instructions.

1     16.    (Original) The computer system of claim 10, wherein the block of

2  instructions to be executed transactionally comprises a critical section.

1     17.    (Original) The computer system of claim 10, wherein the commit

2  instruction is a native machine code instruction of the processor.

1     18.    (Original) The computer system of claim 10, wherein the commit

2  instruction is defined in a platform-independent programming language.

1     19.    (Currently amended) A computer-readable storage medium storing

2  instructions that when executed by a computer cause the computer to perform a

3  method for executing ~~computing means that supports~~ a commit instruction to

4  facilitate transactional execution, ~~wherein the commit instruction marks the end~~

5  ~~of a block of instructions to be executed transactionally,~~ comprising:

6     ~~a processing means; and~~

7     ~~an execution means within the processing means;~~

8     ~~wherein upon encountering the commit instruction, the execution means is~~

9  ~~configured to successfully complete transactional execution of the block of~~

10  ~~instructions preceding the commit instruction;~~

11     ~~wherein changes made during the transactional execution are not~~

12  ~~committed to the architectural state of the processor until the transactional~~

13  ~~execution successfully completes.~~

14     encountering the commit instruction during execution of a program,

15  wherein the commit instruction marks the end of a block of instructions to be

16  executed transactionally; and

17     upon encountering the commit instruction, successfully completing

18  transactional execution of the block of instructions preceding the commit

19  instruction;

20       wherein changes made during the transactional execution are not

21   committed to the architectural state of the processor until the transactional

22   execution successfully completes.


1       20.    (Currently amended) The computer-readable storage medium

2   computing means of claim 19, wherein while successfully completing

3   transactional execution, the execution means is configured to involves:

4       atomically commit committing changes made during the transactional

5   execution; and to

6       resume resuming normal non-transactional execution.